

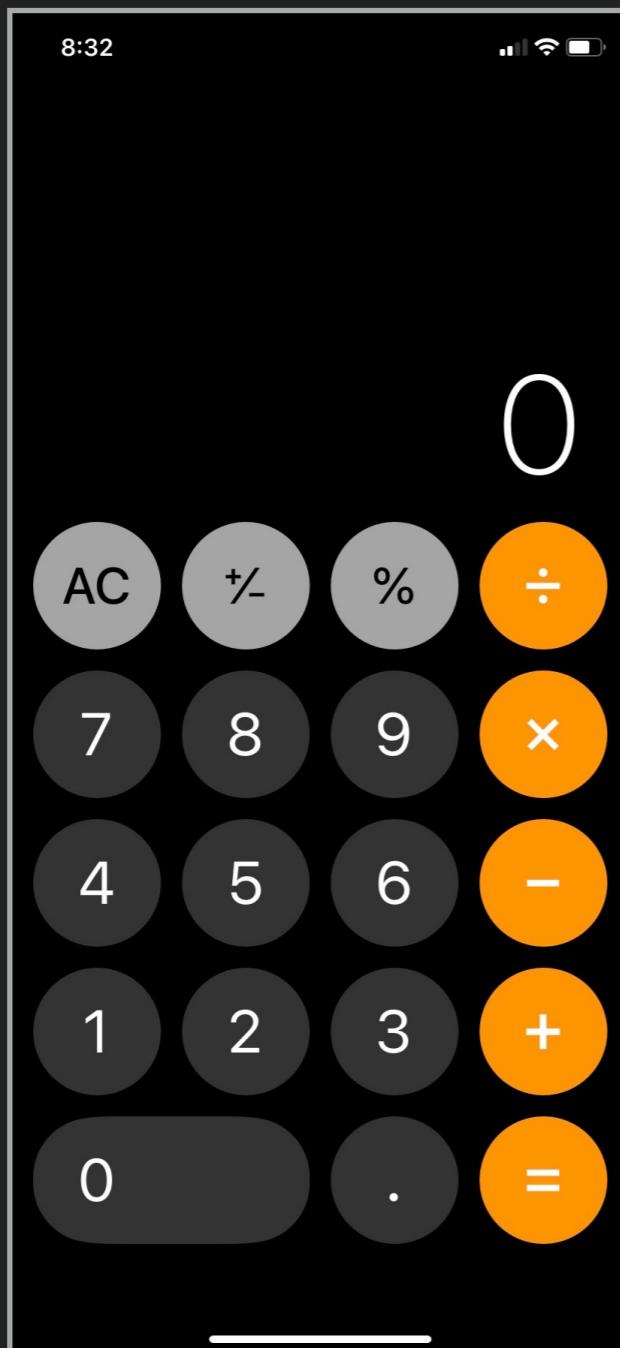
SESSION 3

iOS DEVELOPMENT

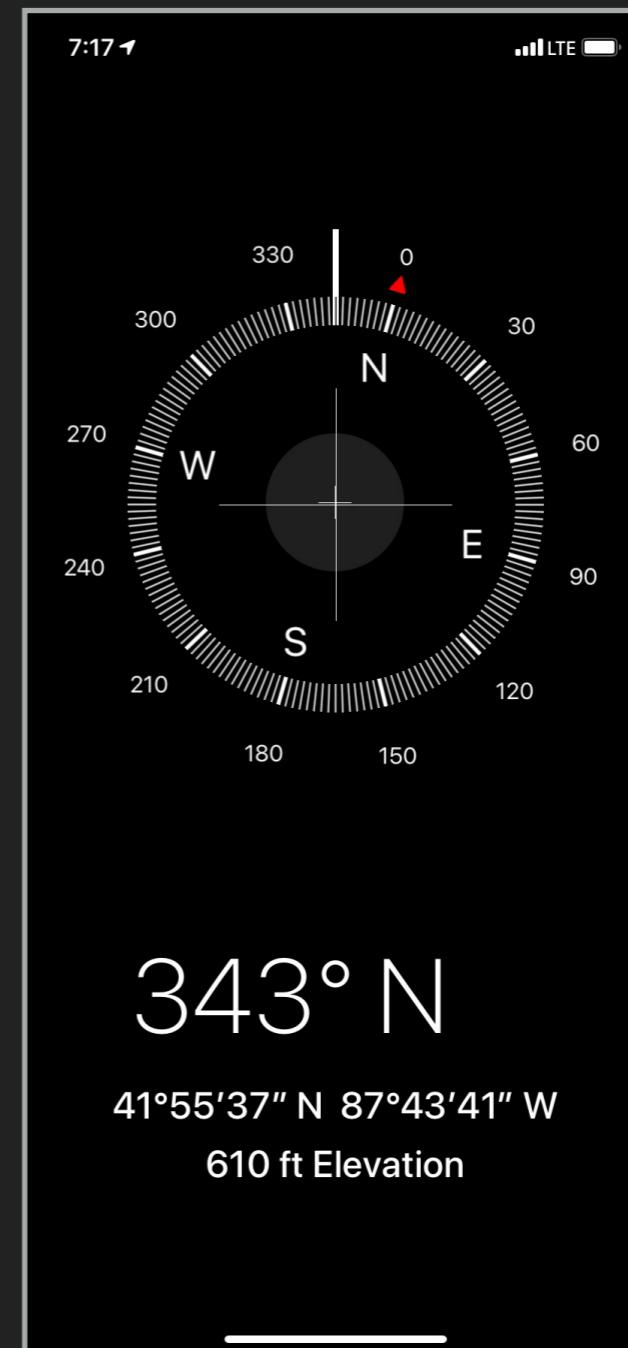
MORE TYPES OF

VIEW CONTROLLERS

APPS CAN HAVE JUST ONE VIEW CONTROLLER...



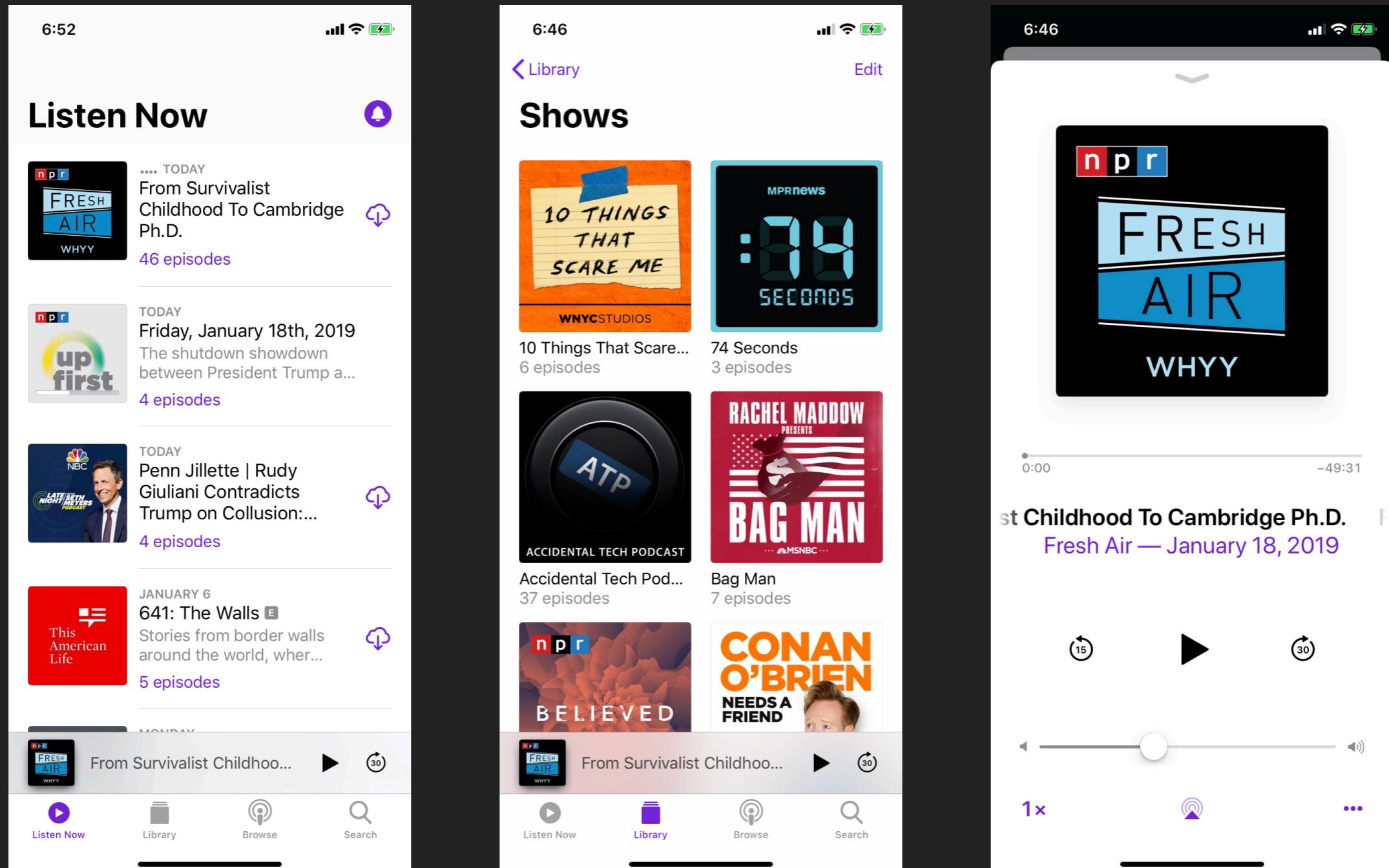
Calculator



Compass

VIEW CONTROLLERS

... OR MULTIPLE VIEW CONTROLLERS



VIEW CONTROLLERS

POP QUIZ*

- ▶ How many view controllers did your assignment have?

Tango



Horse

*Worth zero points.

THE VIEW CONTROLLER FAMILY

- ▶ UITableViewController
- ▶ UICollectionViewController
- ▶ UINavigationController
- ▶ UITabBarController
- ▶ UIPageViewController
- ▶ UISplitViewController

The screenshot shows a mobile application interface with a search bar at the top labeled 'Objects'. Below the search bar is a list of six items, each consisting of an icon and a text description:

- Navigation Controller** - A controller that manages navigation hierarchy of views.
- Table View Controller** - A controller that manages a table view.
- Collection View Controller** - A controller that manages a collection of items.
- Tab Bar Controller** - A controller that manages a set of view controllers that represent tab bar items.
- Split View Controller** - A composite view controller that manages both left and right view controllers.
- Page View Controller** - Presents a sequence of view controllers.

9:41



VIEW CONTROLLERS

Edit

World Clock

7+

TABLE VIEW CONTROLLER

- ▶ Manages a table view
- ▶ Rows of vertically scrolling content

Today, -2HRS

Cupertino

5:21 PM

Today, +1HR

New York

8:21 PM

Tomorrow, +8HRS

Amman

3:21 AM

Tomorrow, +14HRS

Beijing

9:21 AM

Tomorrow, +7HRS

Berlin

2:21 AM

Today, +0HRS

Winnipeg

7:21 PM

Tomorrow, +7HRS

Turin

2:21 AM

Clock

World Clock

Alarm

Bedtime

Stopwatch

Timer

VIEW CONTROLLERS

COLLECTION VIEW CONTROLLER

- ▶ Manages a collection view
- ▶ Rows and columns of vertically or horizontally scrolling content

Podcasts

9:41



Library

Shows

8 Edit



The City
12 episodes



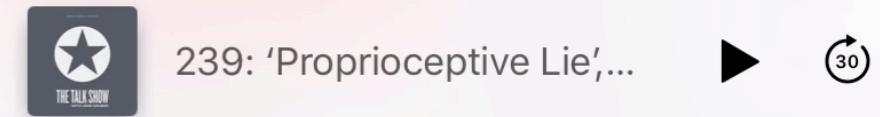
The Daily
87 episodes



The Old Ones with M...
2 episodes



The Talk Show With...
2 episodes



239: 'Proprioceptive Lie',...



30



Listen Now



Library



Browse



Search

NAVIGATION CONTROLLER

- ▶ Navigation bar at the top
- ▶ Manages a stack of child view controllers
- ▶ Coordinates the transitions between child view controllers

Settings

- General
- Control Center
- Display & Brightness
- Wallpaper
- Siri & Search
- Face ID & Passcode
- Emergency SOS
- Battery
- Privacy
- iTunes & App Store
- Wallet & Apple Pay
- Passwords & Accounts
- Mail
- Contacts
- Calendar

VIEW CONTROLLERS

TAB BAR CONTROLLER

- ▶ Tab bar at the bottom
- ▶ Manages an array of child view controllers, displayed one at a time in any order

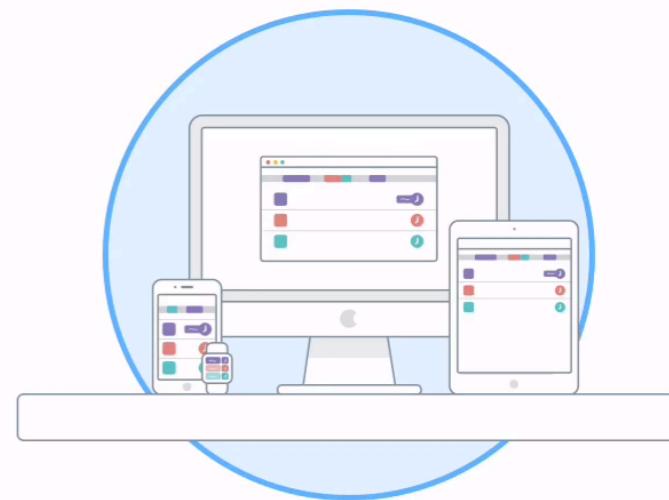
The screenshot shows the Etsy mobile application interface. At the top right, the time is 9:41 and there is a battery icon. The word "Etsy" is in orange. Below the header, there are two tabs: "For you" (selected) and "Etsy Picks". The "For you" tab has a blue underline. The "Etsy Picks" tab is grey. The main content area is titled "Editors' Picks" in bold black text. It features a grid of items: a colorful shelving unit, a small dog figurine, a cat figurine, a grey cat figurine, and a blue plate. To the right of these images, the text "EDITORS' PICKS" and "Design ideas and inspiration" is displayed. Below this section is another grid of items: a small dog figurine, a cat figurine, a grey cat figurine, a small dog photo, and a grey cat photo. To the right of these images, the text "EDITORS' PICKS" and "Personalized and custom jewelry" is displayed. The next section is titled "Shop for gifts" in bold black text. It features a grid of items: a candle holder with two heart-shaped candlesticks, a blue and white plate, and a blue and white bowl. To the right of these images, the text "Vintage home decor" and "Best of winter: under \$30" is displayed. At the bottom of the screen is a tab bar with five icons: "Home" (selected), "Favorites", "Search", "You", and "Cart". The "Home" icon is orange.

PAGE VIEW CONTROLLER

- ▶ Page control at bottom
- ▶ Manages an array of child view controllers, displayed one at a time in sequential order

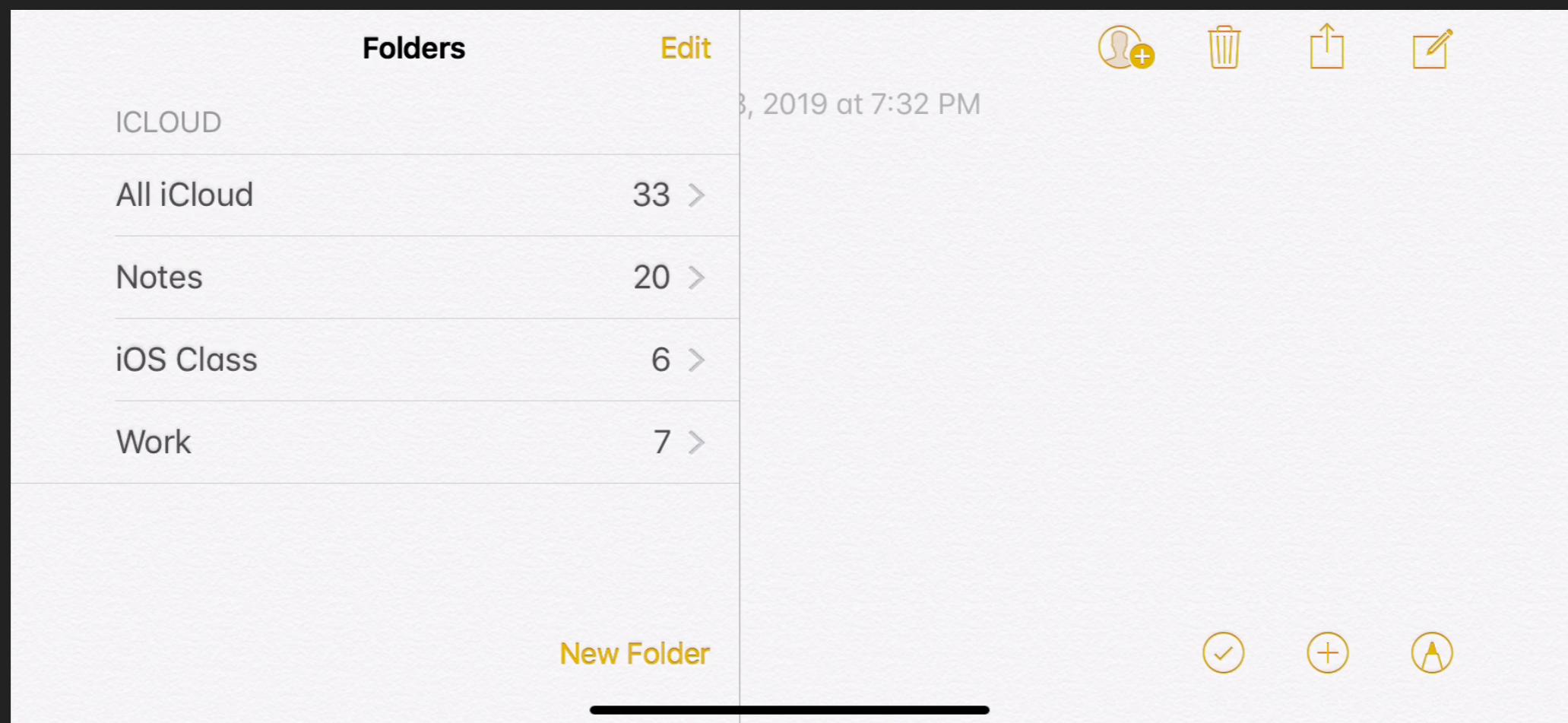
Welcome to Hours

Hours is the time tracker you will actually use.
Swipe to learn how.



SPLIT VIEW CONTROLLER

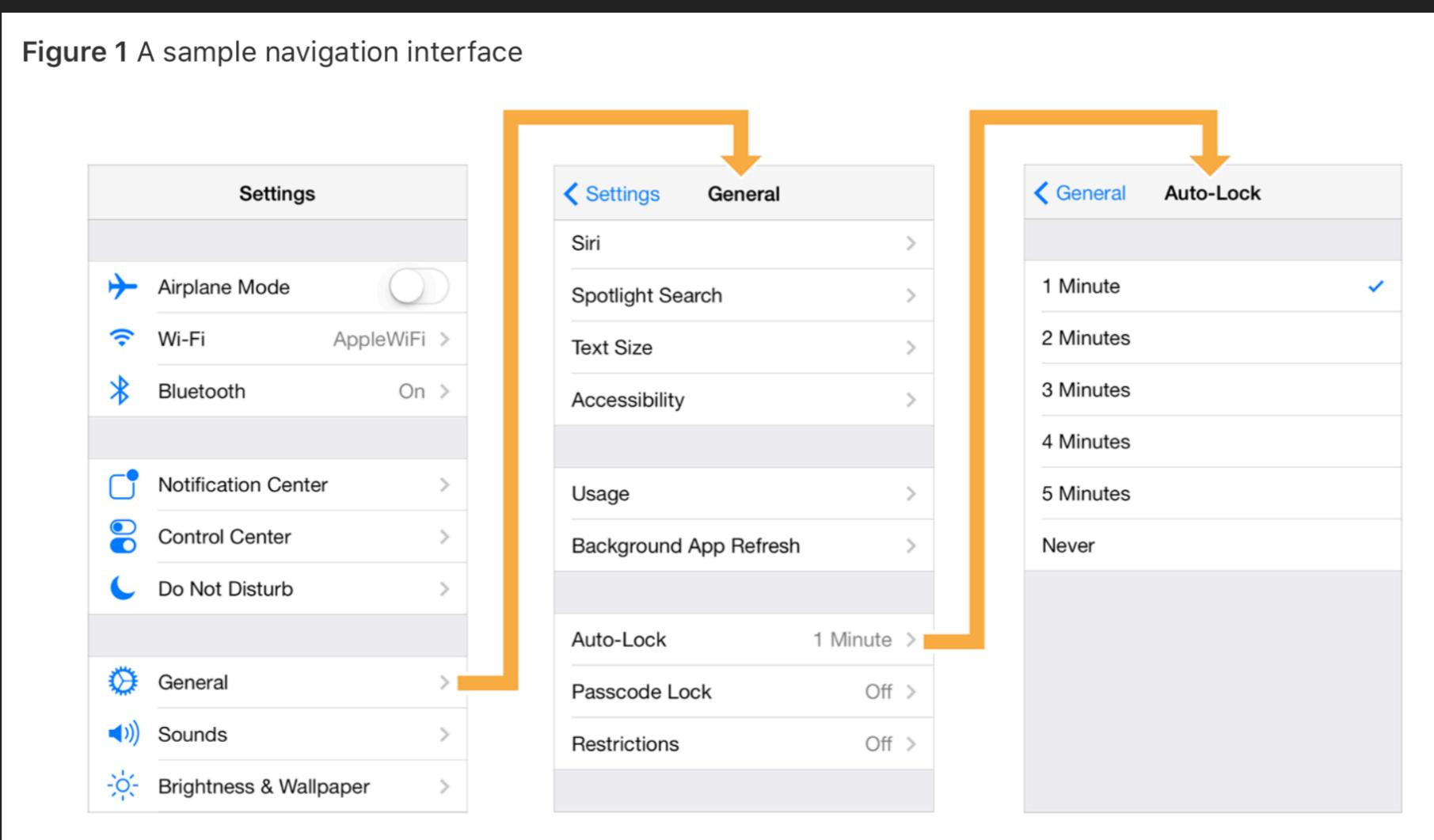
- ▶ Manages a master view controller (left) and a detail view controller (right)



NAVIGATION CONTROLLER

NAVIGATION CONTROLLER

- ▶ A container view controller that defines a **stack-based** scheme for navigating hierarchical content.



NAVIGATION CONTROLLER

- ▶ The navigation controller maintains an ordered array of child view controllers
 - ▶ This is called the **navigation stack**
 - ▶ Root view controller is at the bottom
 - ▶ Currently visible controller is at the top

NAVIGATION CONTROLLER

NAVIGATION BAR

- ▶ Navigation controllers also manage a navigation bar, consisting of:
 - ▶ Left bar button item(s)
 - ▶ Title view
 - ▶ Right bar button item(s)

◀ Library

Podcasts



Conan O'Brien Needs
A Friend
Sunday

3



Fresh Air
Yesterday

4



Late Night with Seth
Meyers Podcast
Wednesday

1



Life Kit
Wednesday

3



The Anthropocene
Reviewed
Dec 26, 2019



This American Life
Sunday

2



Up First
Today

1



Wait Wait... Don't
Tell Me!
Today

1

NAVIGATION BAR

- ▶ Child view controllers tell the navigation controller what to display in the navigation bar
 - ▶ This is done through `UIViewController`'s `navigationItem` and `title` properties

`UIViewController`

```
var navigationItem: UINavigationItem?
```

```
var title: String?
```

```
...
```

NAVIGATION CONTROLLER

LEFT BAR BUTTON ITEM

- ▶ Typically provides navigation back to the previous view controller
 - ▶ Instance of **UIBarButtonItem**
 - ▶ By default, displays the title of the previous view controller

1:58

5G

Library

Podcasts



Conan O'Brien Needs
A Friend
Sunday



Fresh Air
Yesterday

4



Late Night with Seth
Meyers Podcast
Wednesday

1



Life Kit
Wednesday

3



The Anthropocene
Reviewed
Dec 26, 2019



This American Life
Sunday

2



Up First
Today

1



Wait Wait... Don't
Tell Me!
Today

Castaway

LEFT BAR BUTTON ITEM

- ▶ To customize the left bar button item...
 - ▶ On the current view controller, set:
 - ▶ `navigationItem.leftBarButtonItem` -OR-
 - ▶ `navigationItem.leftBarButtonItems`
 - ▶ On the previous view controller, set:
 - ▶ `navigationItem.backBarButtonItem`

NAVIGATION CONTROLLER

LEFT BAR BUTTON ITEM

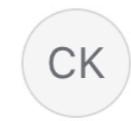
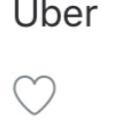
- ▶ Example of a custom left bar button item



Rachel Anderson paid Jalon Nielson
1m



Brandon Weis charged Austin Williams
1m



Christy Kelly paid Julia Irle
1m

you're actually the best



Elsa Jimenez paid Leobardo Fimbres
1m

Gracias por tu compañía



Madison Sams charged Connor Alvin
1m



Venmo



Alley Colecchia paid Olivia Terlizzi
1m

NAVIGATION CONTROLLER

TITLE VIEW

- ▶ By default, displays a label with the view controller's **title** property

Castaway

1:58

5G

Library

Podcasts



Conan O'Brien Needs
A Friend
Sunday

3



Fresh Air
Yesterday

4



Late Night with Seth
Meyers Podcast
Wednesday

1



Life Kit
Wednesday

3



The Anthropocene
Reviewed
Dec 26, 2019



This American Life
Sunday

2



Up First
Today

1

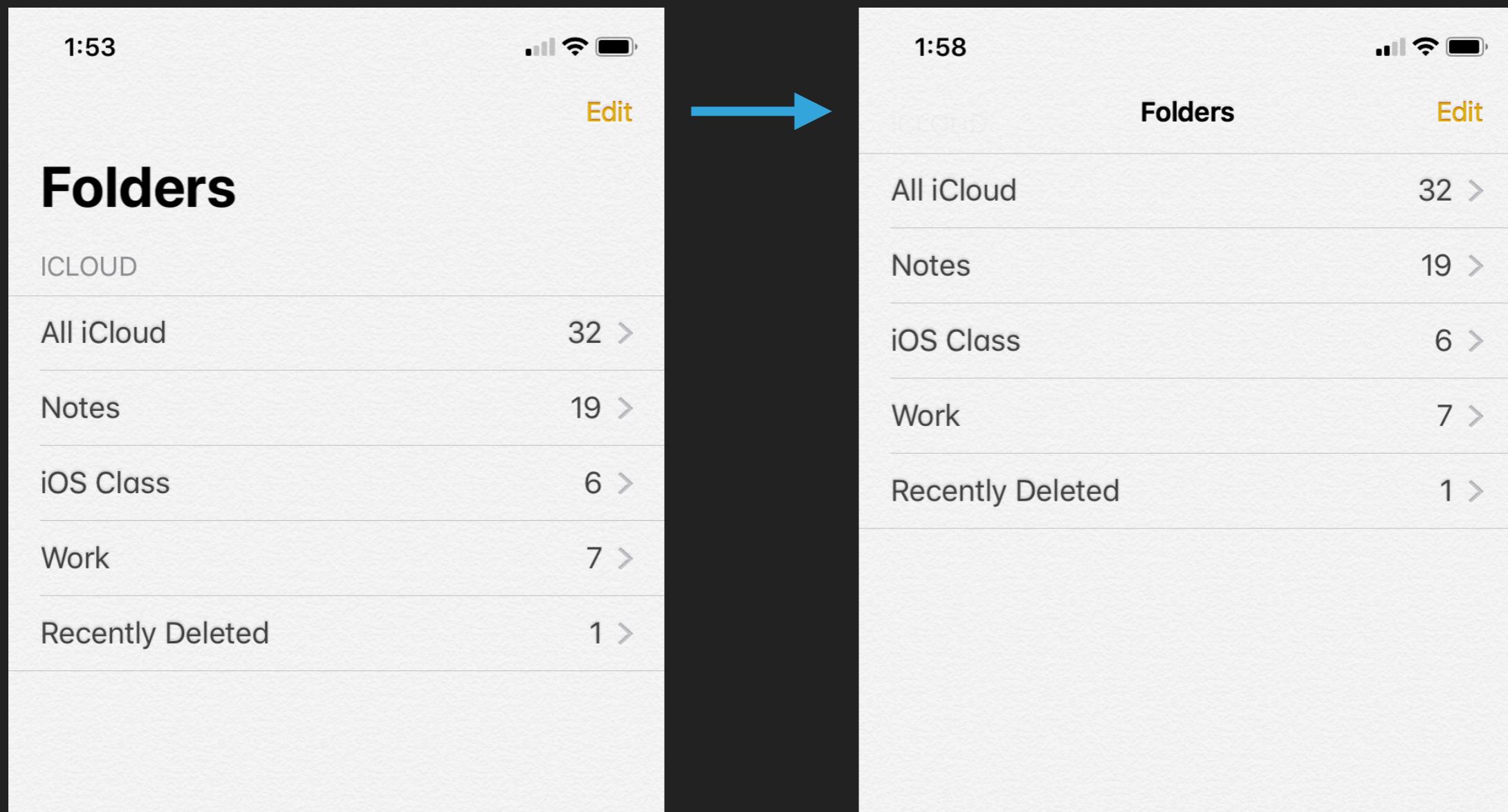


Wait Wait... Don't
Tell Me!
Today

1

LARGE TITLES

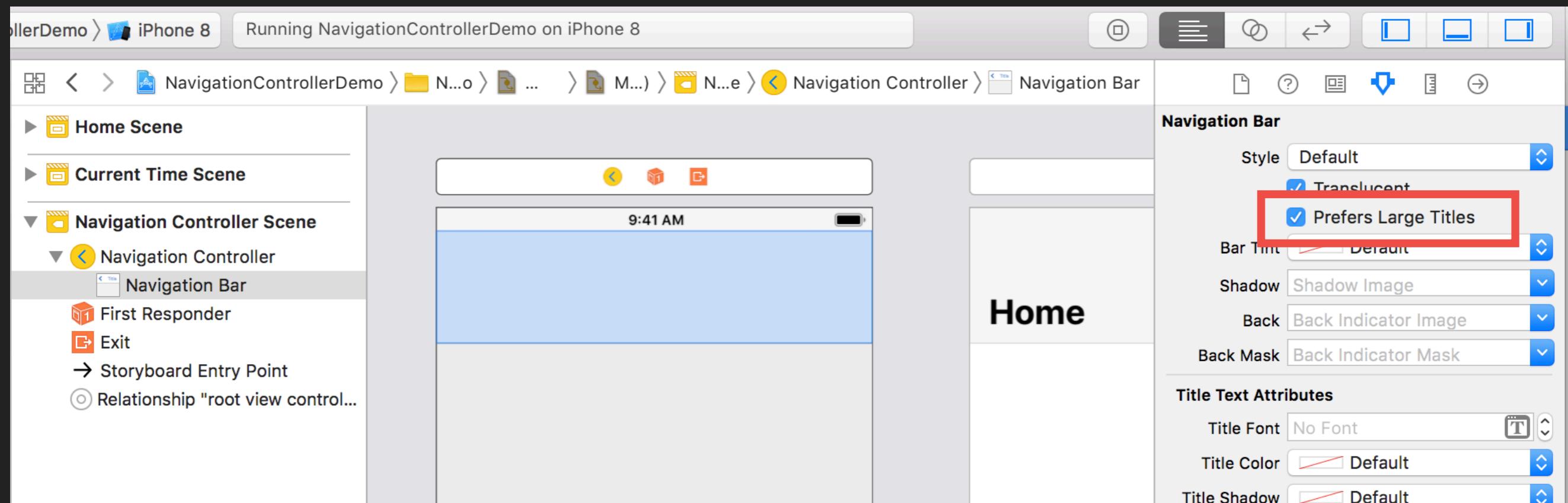
- ▶ Large titles were introduced in iOS 11



LARGE TITLES

- ▶ Select navigation bar and check “Prefers Large Titles”
- ▶ Can also be done programmatically:

```
navigationController?.navigationBar.prefersLargeTitles = true
```

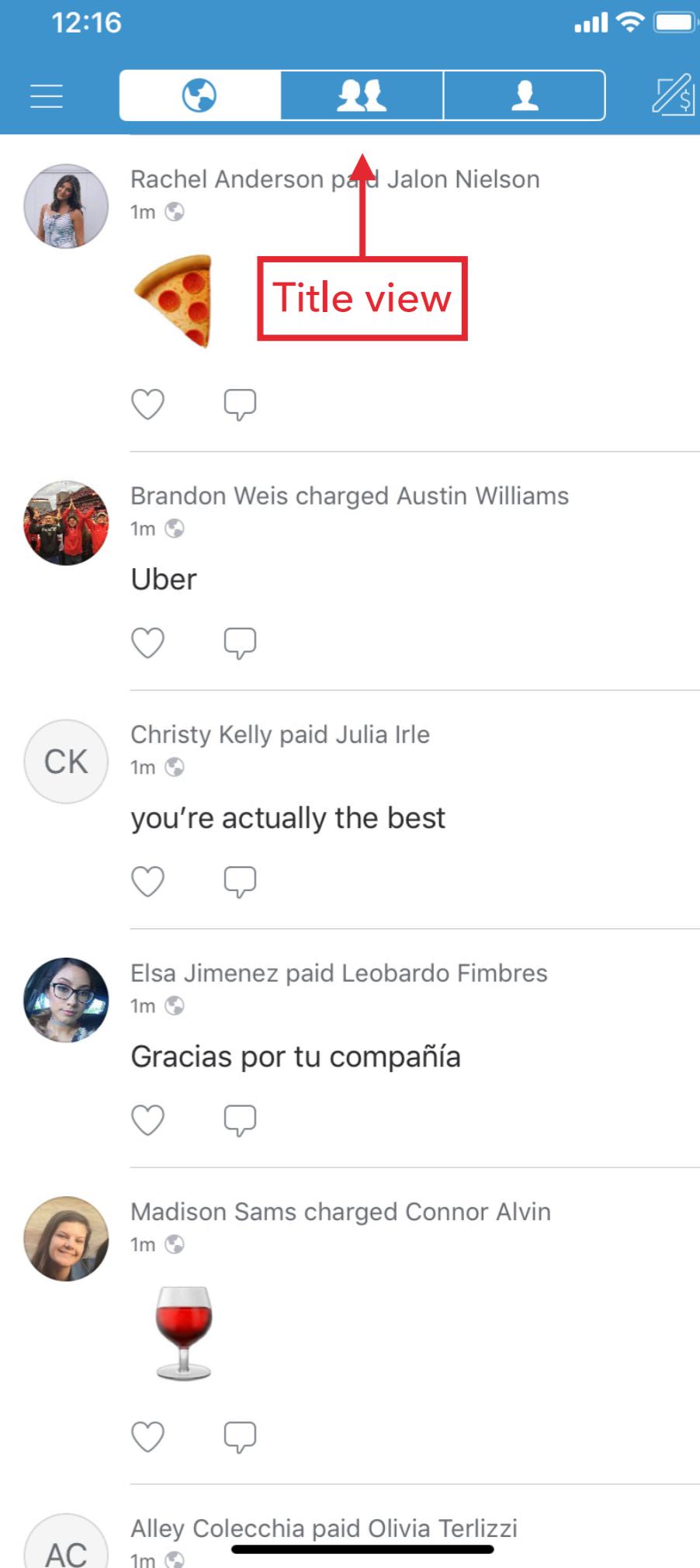


NAVIGATION CONTROLLER

TITLE VIEW

- ▶ You can provide a custom title view by setting `navigationItem.titleView`

Venmo



NAVIGATION CONTROLLER

RIGHT BAR BUTTON ITEM

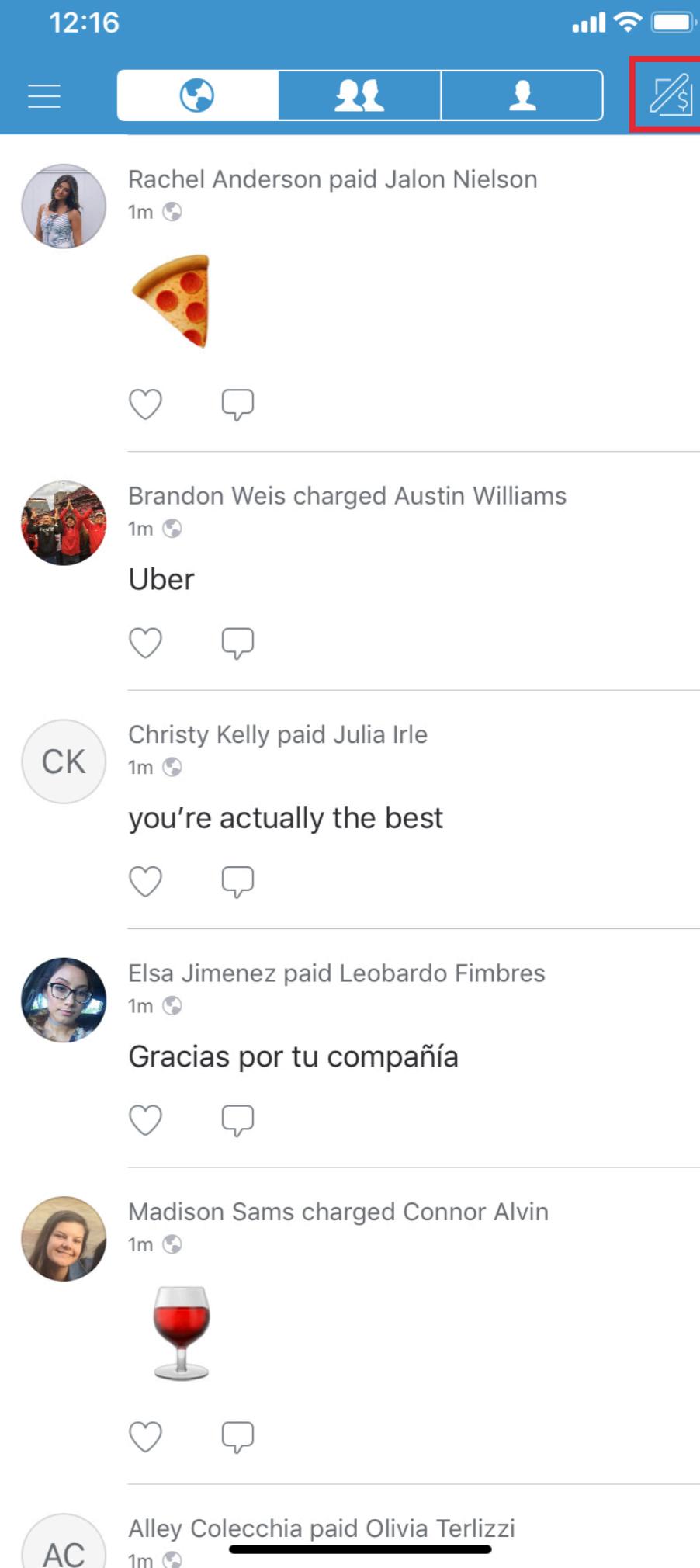
- ▶ Set right bar button item(s) with:

`navigationItem.rightBarButtonItem`

-OR-

`navigationItem.rightBarButtonItems`

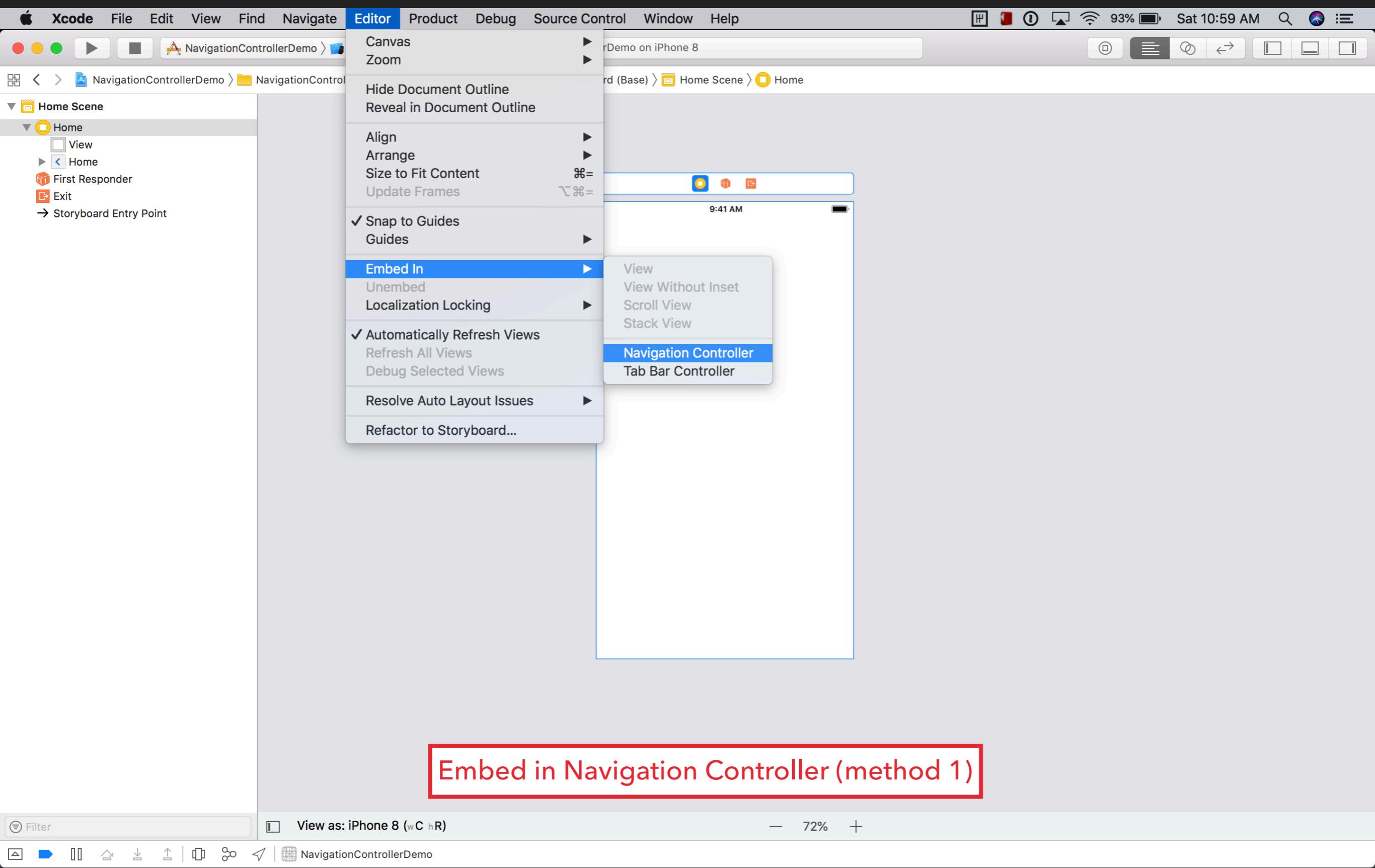
Venmo

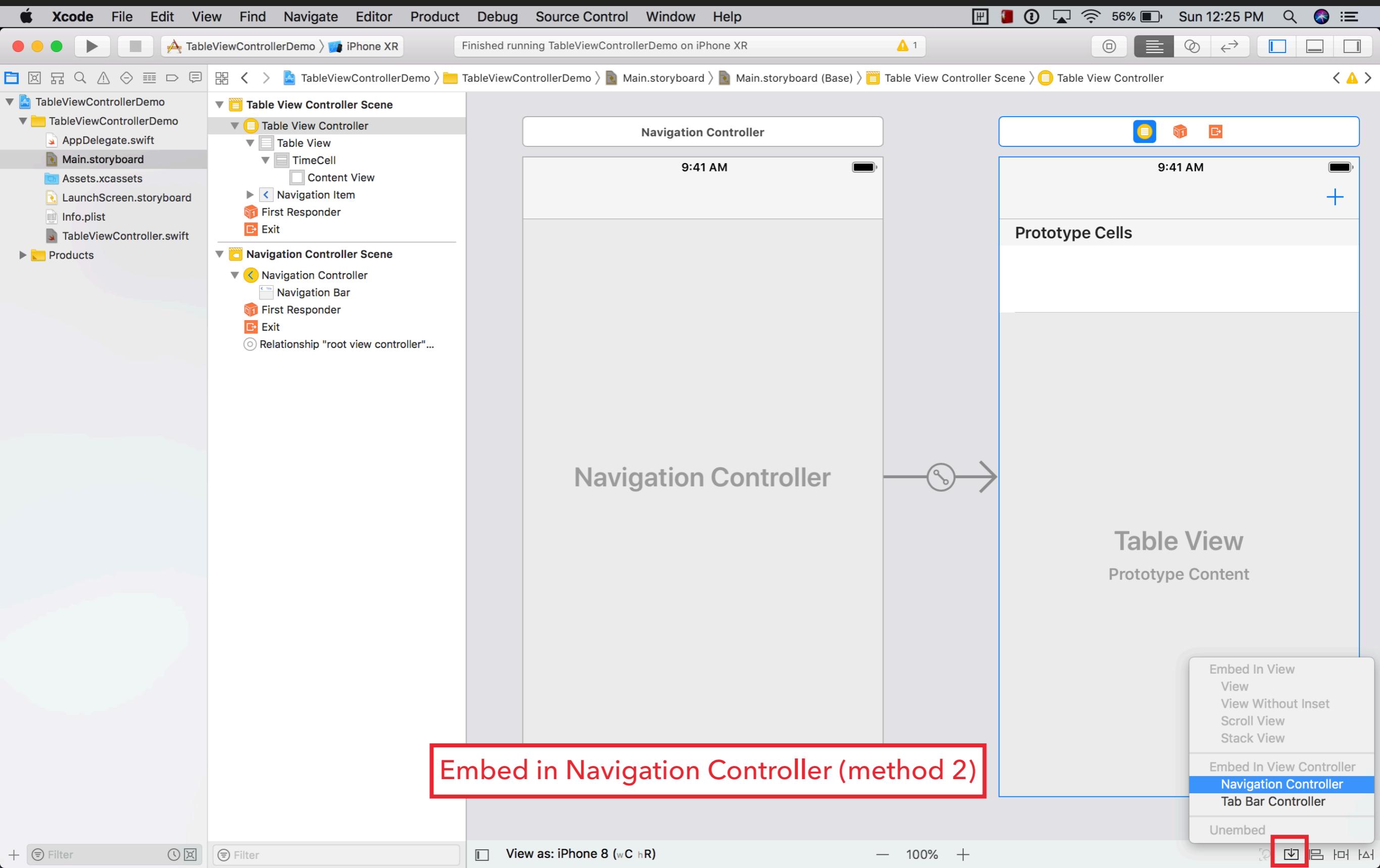


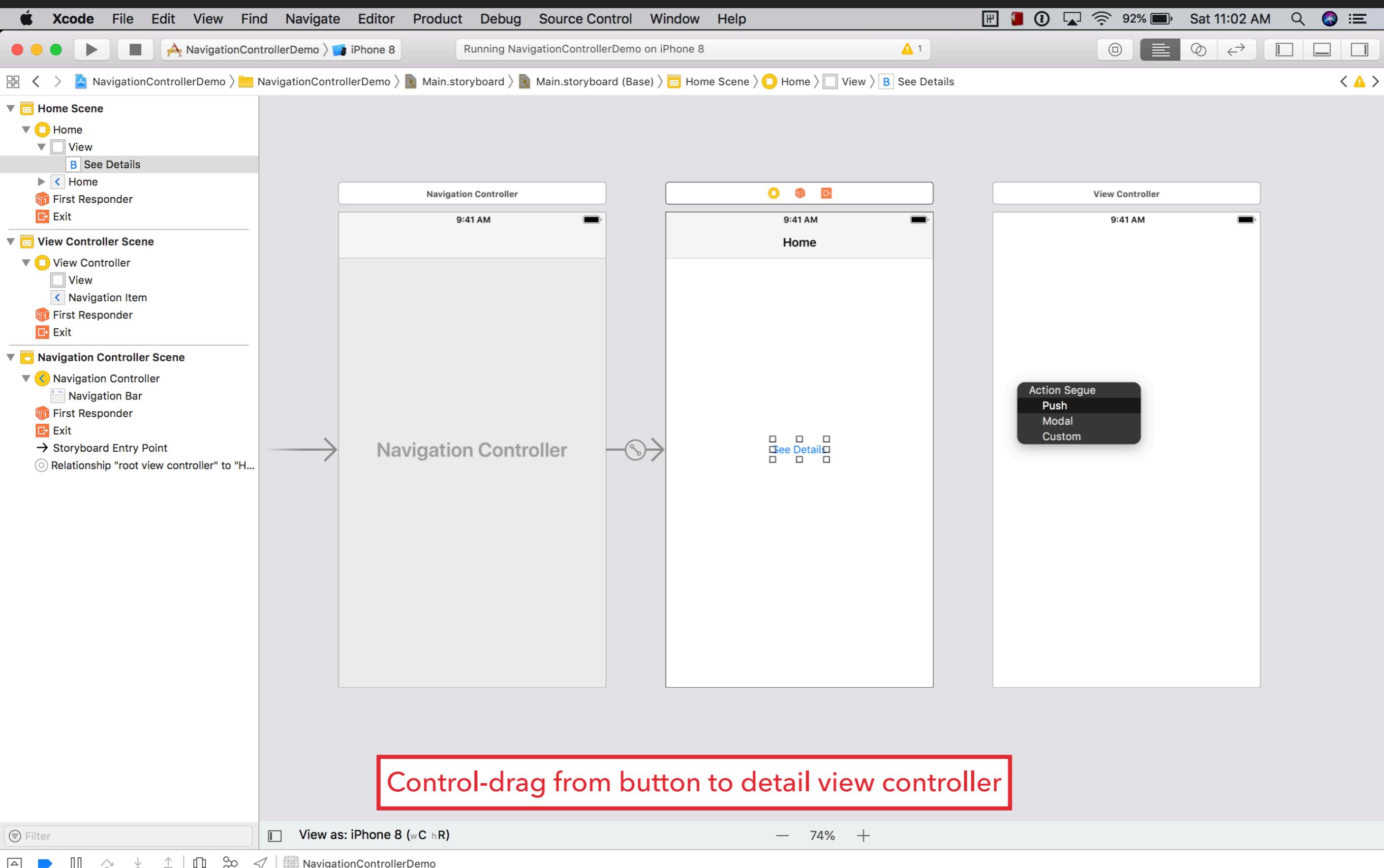
CREATING A NAVIGATION STACK

- ▶ The easiest way to create a navigation stack is using storyboard
- ▶ **Segues*** establish relationships and transitions between view controllers

*Pronounced “segway”.

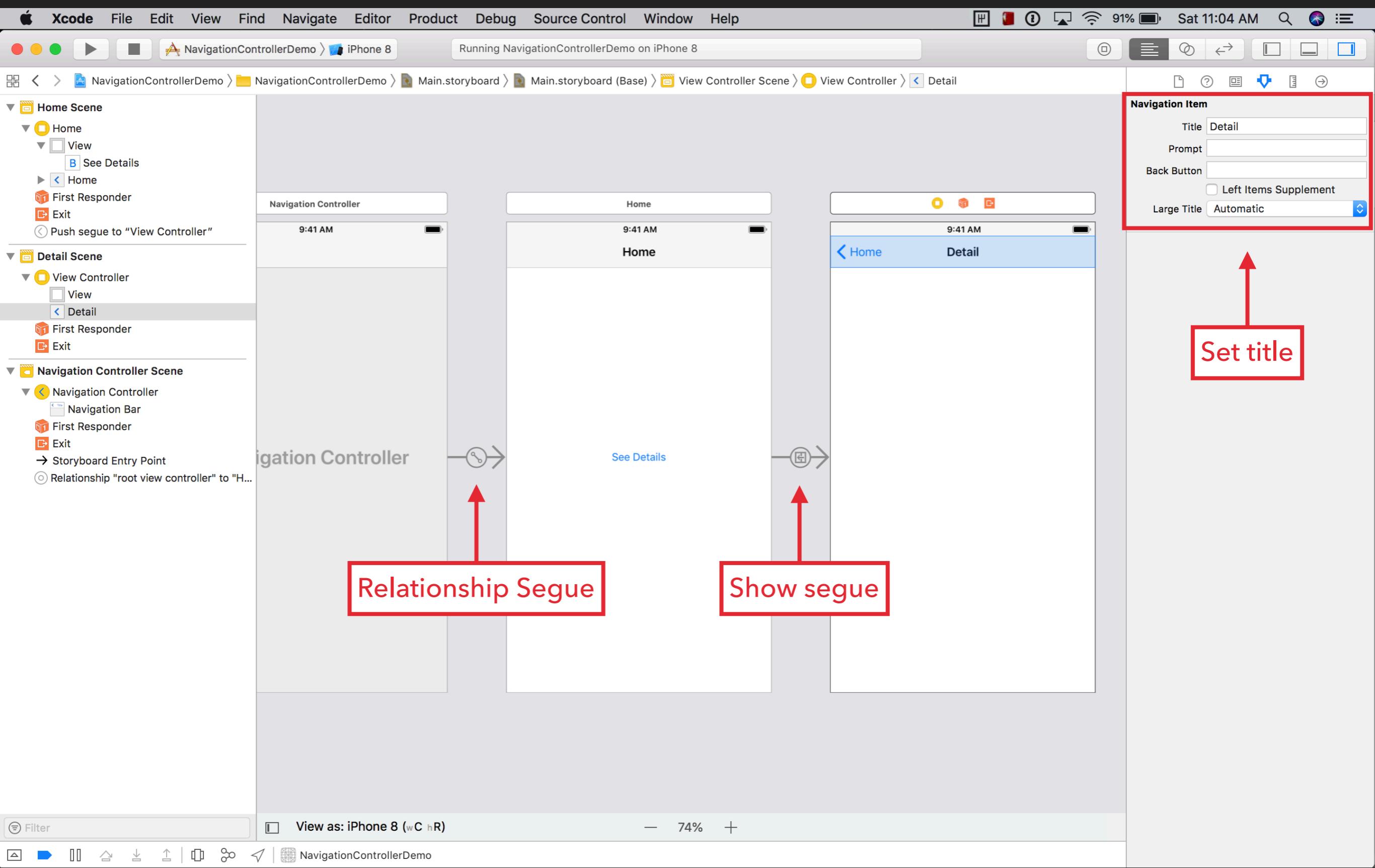






NAVIGATION CONTROLLER

30



PREPARE FOR SEGUE

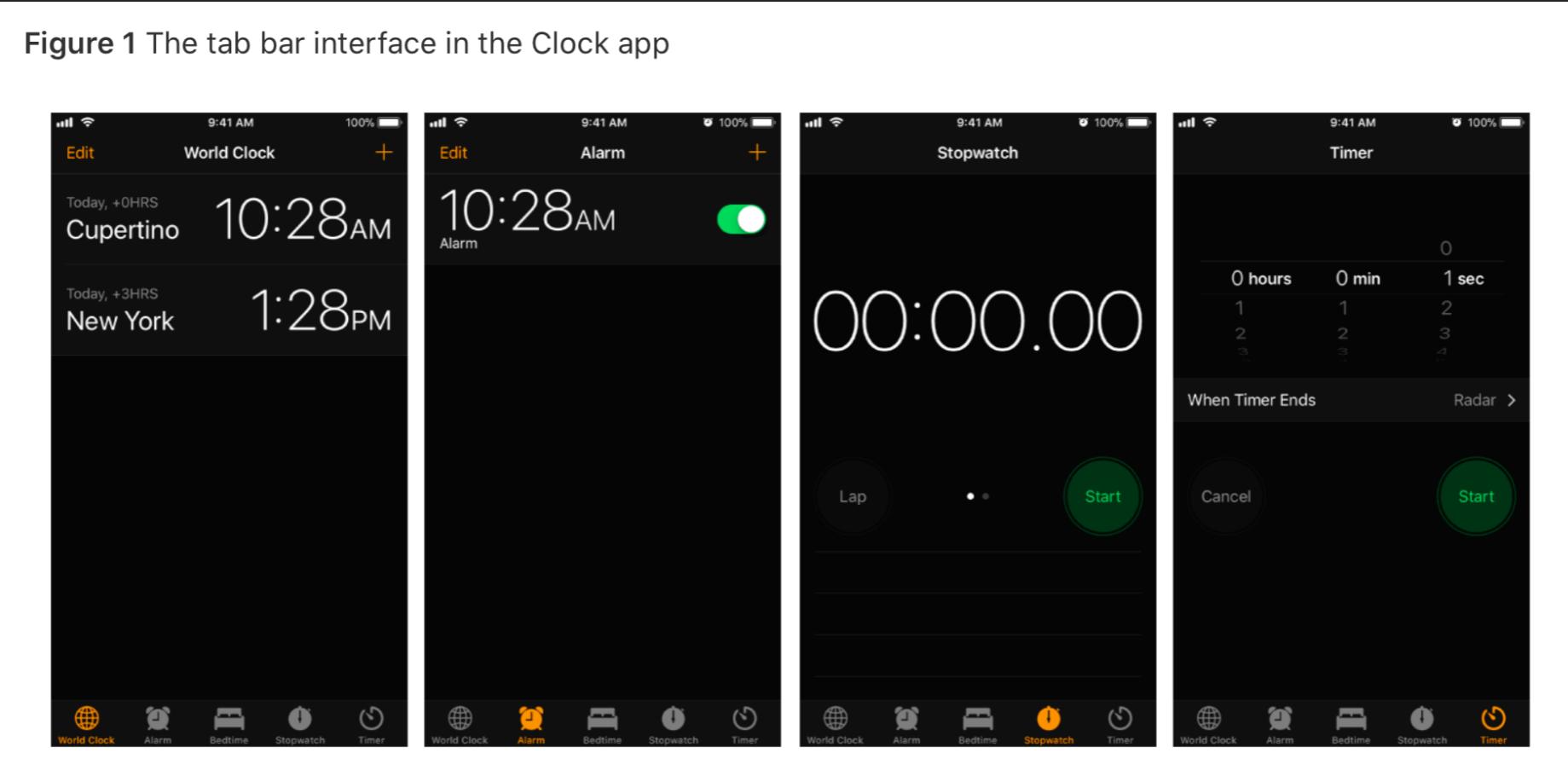
- ▶ Override UIViewController's "prepare for segue" method to pass data between view controllers before the transition occurs

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {  
    let destination = segue.destination as? TimeViewController  
    let formatter = DateFormatter()  
    formatter.timeStyle = .long  
    destination?.currentTime = formatter.string(from: Date())  
}
```

TAB BAR CONTROLLER

TAB BAR CONTROLLER

- ▶ A container view controller that manages a **radio-style selection** interface, where the selection determines which child view controller to display.



TAB BAR

- ▶ Child view controllers tell the tab bar controller what to display in the tab bar
 - ▶ This is done through UIViewController's **tabBarItem** and **title** properties

UIViewController

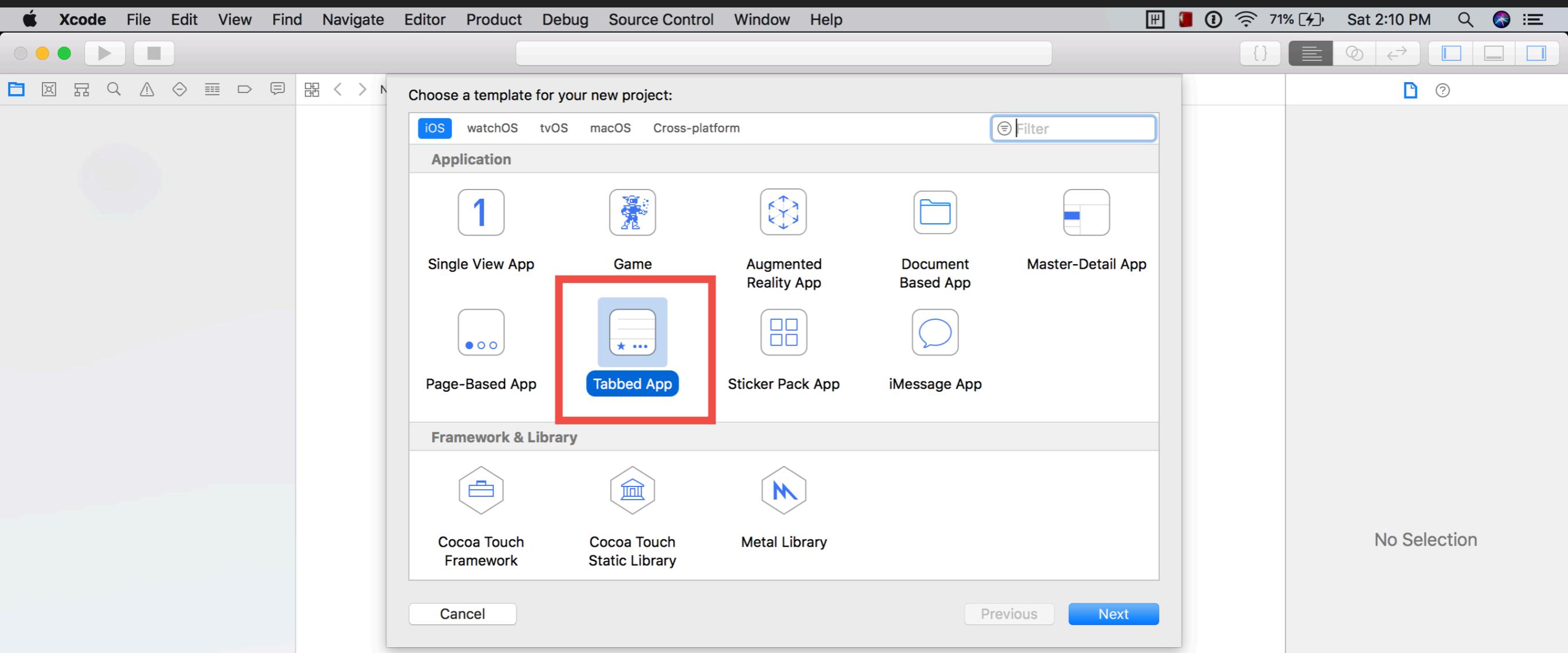
```
var tabBarItem: UITabBarItem?
```

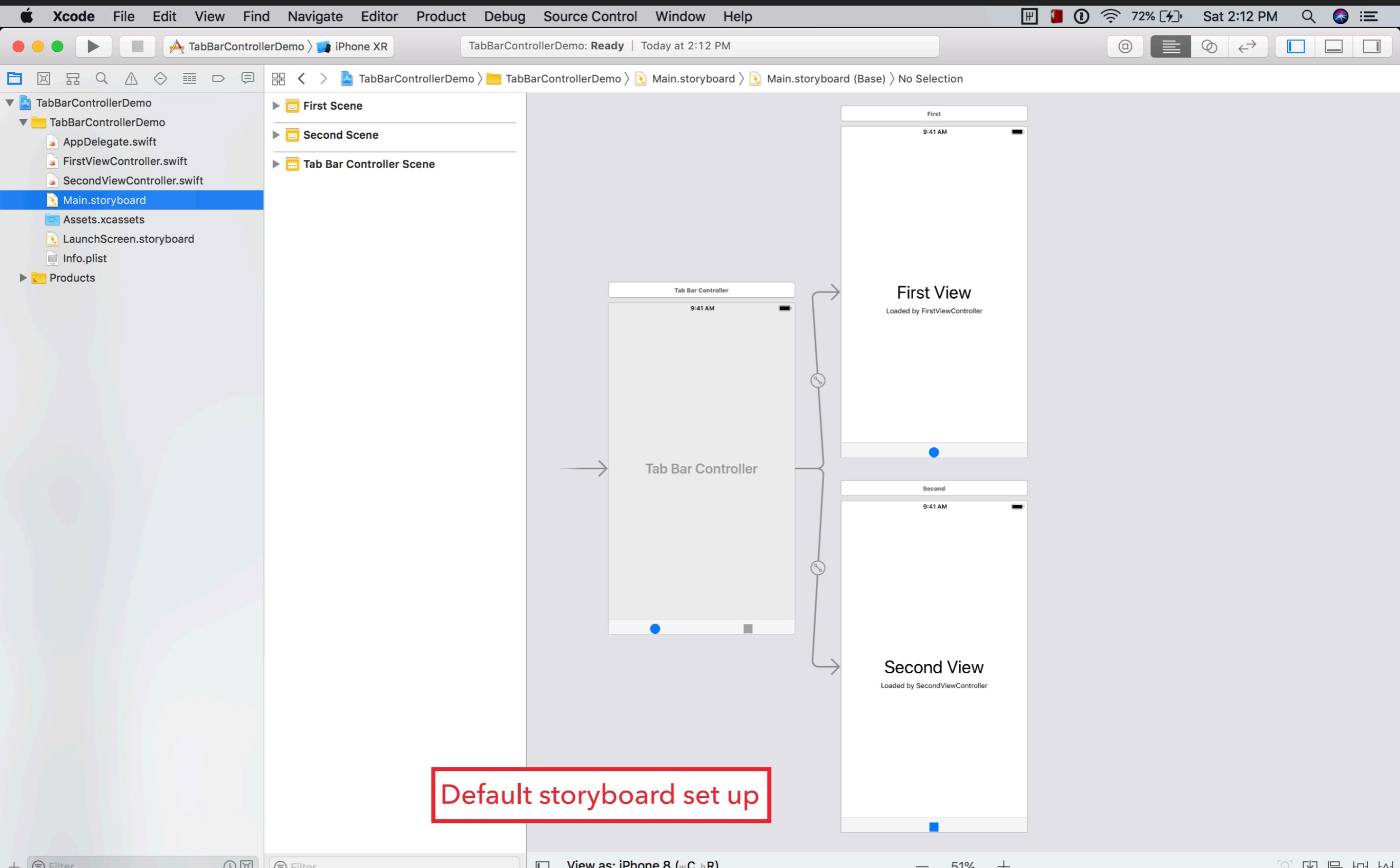
```
var title: String?
```

```
...
```

TAB BAR ITEM

- ▶ Each `UITabBarItem` can have a title and image
 - ▶ Use `UITabBarItemAppearance` to specify whether title and image should be stacked vertically or side-by-side





TAB BAR CONTROLLER

Genres

Featured

Charts

38

TAB BAR ITEMS

- “More” tab will automatically be displayed if you have more than 5 tabs



This tender adaptation is exquisitely acted and dreamily atmospheric.

Top TV Shows

	PAW Patrol Mighty Pups		Schitt's Creek Schitt's Cre...		Dirty John: The Dirty Tr...		Surviving R. Kelly Season 1
--	---------------------------	--	-----------------------------------	--	--------------------------------	--	-----------------------------------

See All >

Latest TV Episodes

	Break On Through Hell's Kitchen, Season 18	\$2.99
	A Champagne Mood Dynasty, Season 2	\$2.99
	Driver's Eddie 2: Orlando... Fresh Off the Boat, Season 5	\$2.99
	H--HEY, YOU Speechless, Season 3	\$2.99

See All >



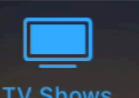
DISCOVER A
GREAT SERIES

Seasons from \$9.99

New & Noteworthy

See All >

iTunes Store



TV Shows



Tones



Downloads

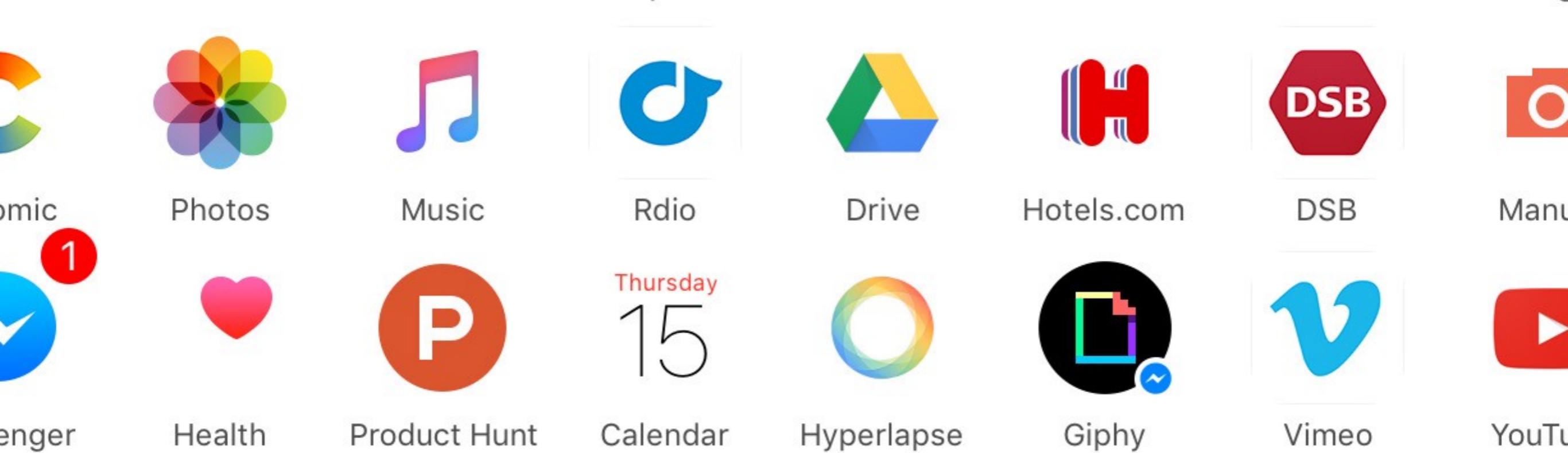


Genius



More

TABLE VIEW CONTROLLER



<https://medium.com/swlh/let-s-talk-about-white-app-icons-ce2e83b9eb86>

THERE ARE 2 MILLION+ APPS IN THE APP STORE

WHAT DO THEY HAVE IN COMMON?

**THEY ALL HAVE
TABLE VIEWS**

(Okay, almost all.)

TABLE VIEW CONTROLLER

WHAT IS A TABLE VIEW?

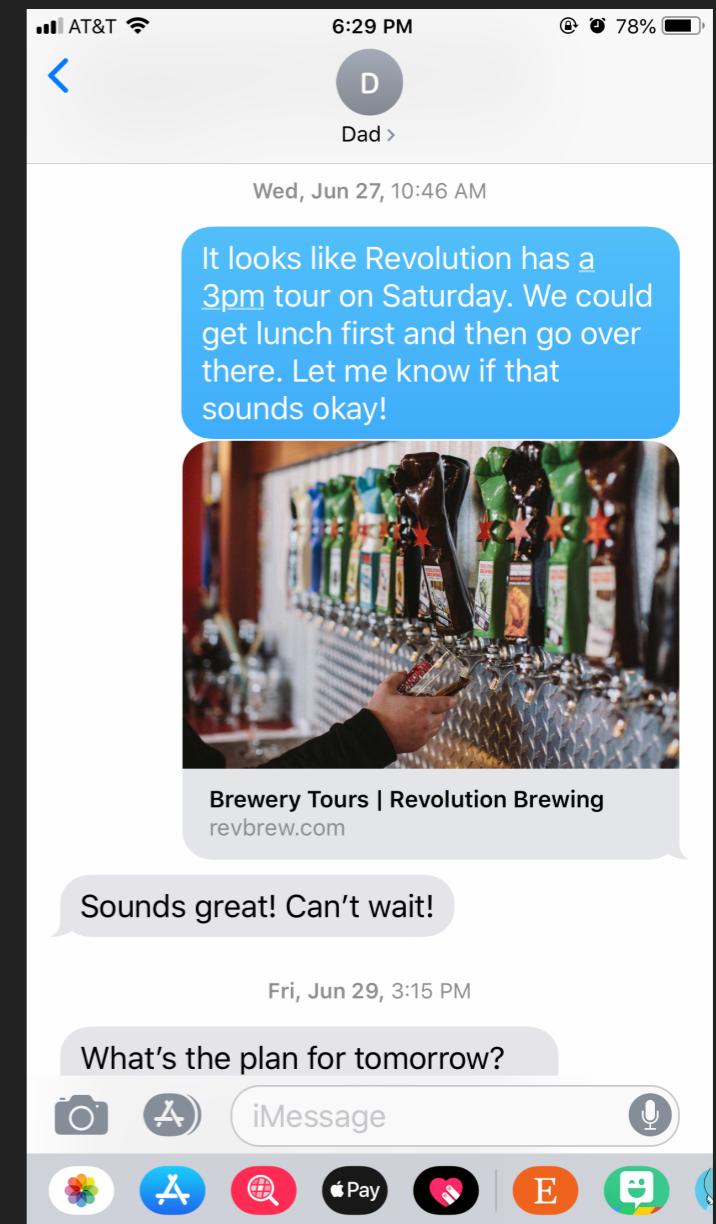
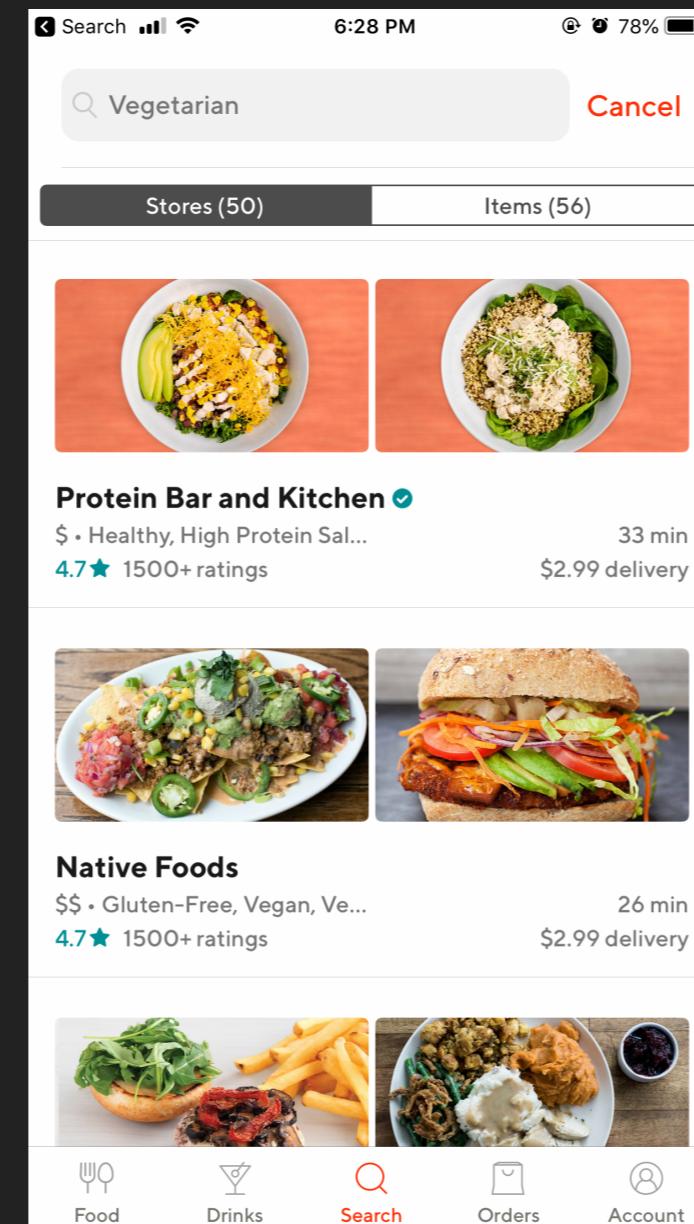
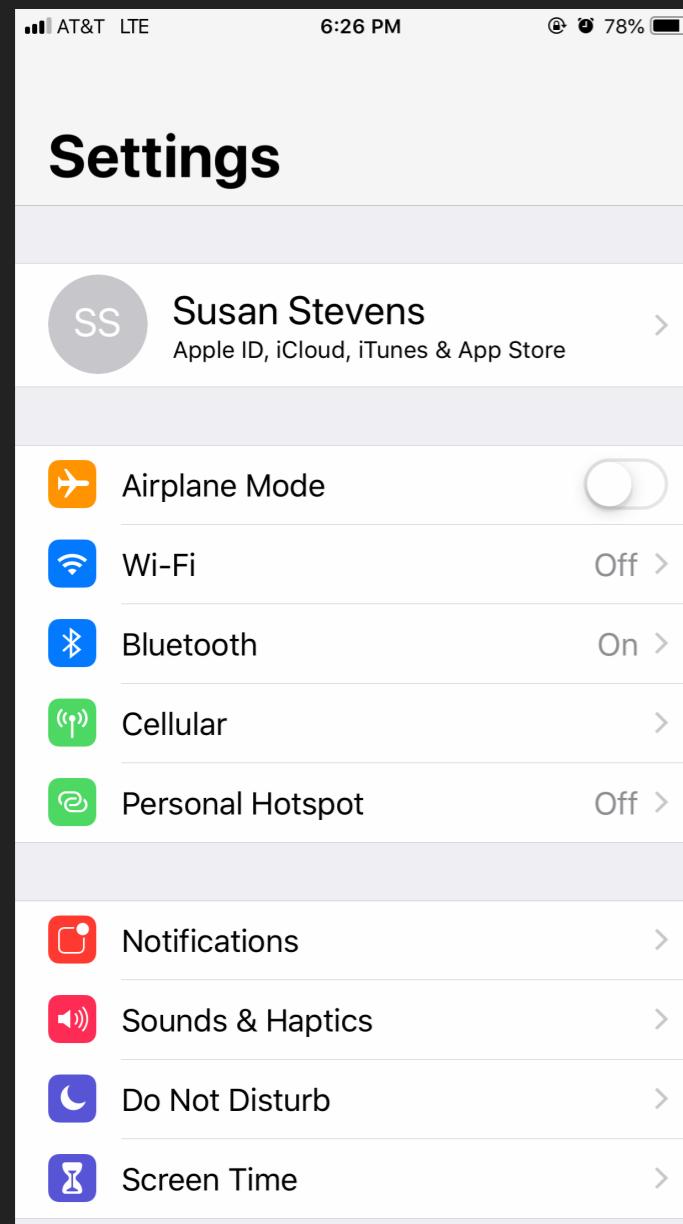
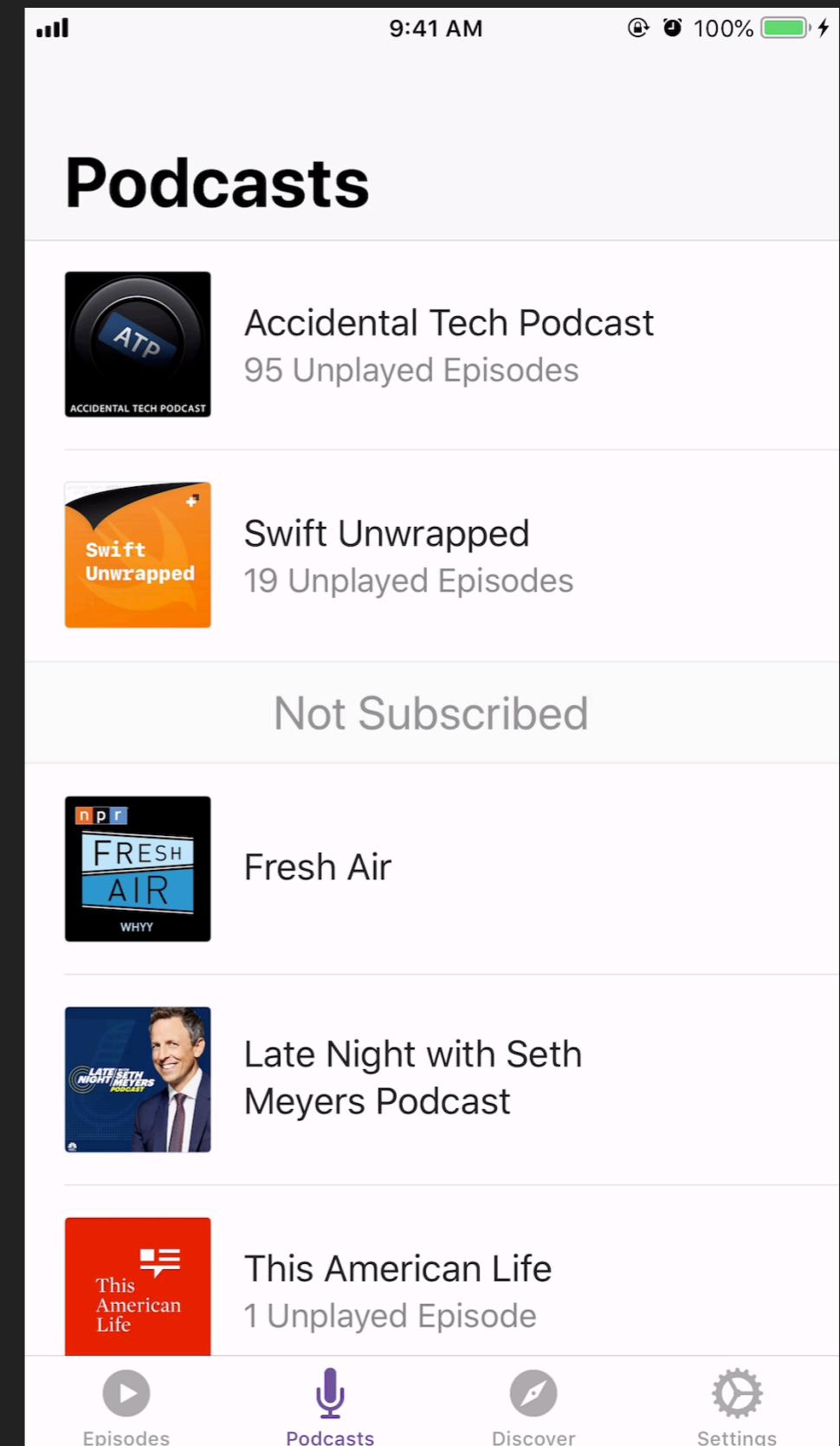


TABLE VIEW CONTROLLER

WHAT IS A TABLE VIEW?

- ▶ If it has rows that scroll vertically, it's probably a **UITableView**
- ▶ Beyond that, UITableViews are infinitely customizable



HOW DOES IT WORK?

- ▶ Apple does all of the hard work (scrolling, rendering) so that you can do the fun stuff. 😊
- ▶ You conform to two protocols:
 - ▶ `UITableViewDataSource`
 - ▶ `UITableViewDelegate`

TABLE VIEW DATA SOURCE

- ▶ At minimum, `UITableView` needs to know three things:
 - ▶ How many sections should it have?
 - ▶ How many rows in each section?
 - ▶ Which cell should be displayed in each row?

TABLE VIEW DATA SOURCE

- ▶ You answer these questions with three `UITableViewDataSource` methods

```
24
25  override func numberOfSections(in tableView: UITableView) -> Int {
26      return 1
27  }
28
29  override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
30      return 20
31  }
32
33  override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
34      let cell = tableView.dequeueReusableCell(withIdentifier: "reuseIdentifier", for: indexPath)
35
36      // Configure the cell...
37
38      return cell
39  }
```

TABLE VIEW DATA SOURCE

- ▶ Imagine a table view with thousands of rows.
- ▶ Would you want to load every row into memory as soon as the table view appears on screen?

TABLE VIEW DATA SOURCE

- ▶ Instead of creating every cell, table views maintain only the cells they need at a given time
- ▶ As the user scrolls, some cells go off screen and others come on screen
- ▶ Rather than creating entirely new cells, table views **reuse** cells as the user scrolls

TABLE VIEW DATA SOURCE

- ▶ The table view will ask the data source for a cell to display at a particular index path (row and section)

```
52  
53 override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {  
54     let cell = tableView.dequeueReusableCell(withIdentifier: "reuseIdentifier", for: indexPath)  
55  
56     // Configure the cell...  
57  
58     return cell  
59 }
```

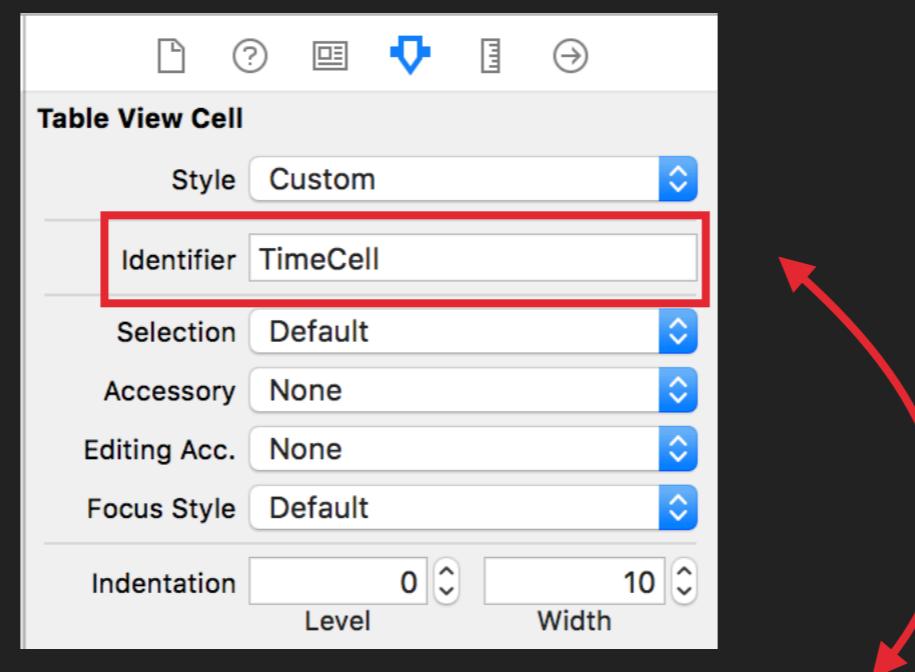
TABLE VIEW DATA SOURCE

- ▶ The data source gets a reusable cell from the table view using the **reuse identifier**

```
52  
53 override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {  
54     let cell = tableView.dequeueReusableCell(withIdentifier: "reuseIdentifier", for: indexPath)  
55  
56     // Configure the cell...  
57  
58     return cell  
59 }
```

TABLE VIEW DATA SOURCE

- ▶ Remember to set the reuseIdentifier in storyboard



```
override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {  
    let cell = tableView.dequeueReusableCell(withIdentifier: "TimeCell", for: indexPath)  
    cell.textLabel?.text = timeStrings[indexPath.row]  
    return cell  
}
```

TABLE VIEW DATA SOURCE

- ▶ The data source configures the cell for the given index path and then returns it

```
52  
53 override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {  
54     let cell = tableView.dequeueReusableCell(withIdentifier: "reuseIdentifier", for: indexPath)  
55  
56     // Configure the cell...  
57  
58     return cell  
59 }
```

TABLE VIEW CONTROLLER

TABLE VIEW CONTROLLER

- ▶ Requires less set-up
- ▶ Built-in support for editing and refreshing
- ▶ Table view fills the entire view controller

Table View Controller

53

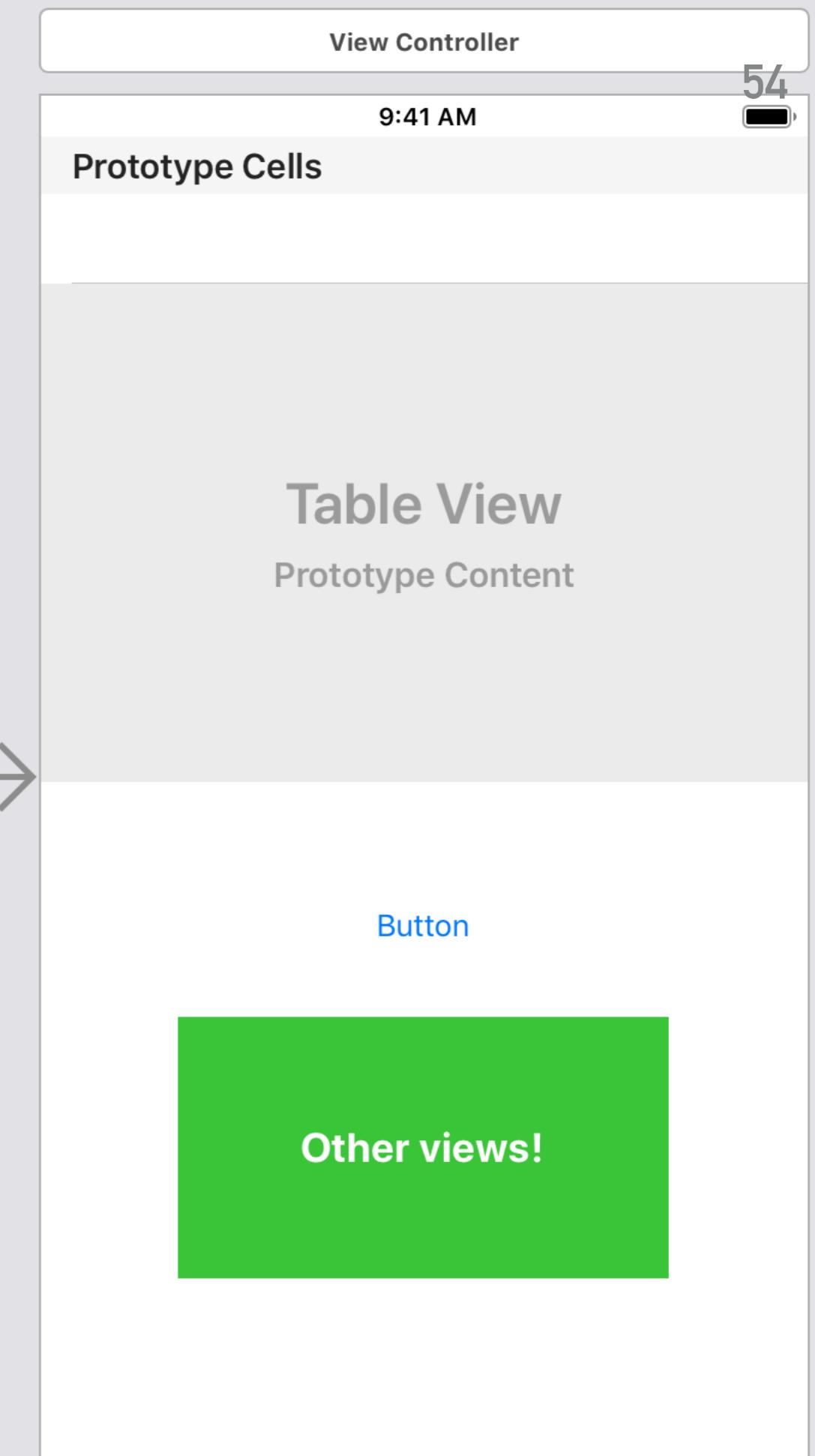
9:41 AM

Prototype Cells

Table View
Prototype Content

VIEW CONTROLLER + TABLE VIEW

- ▶ Requires more set-up
- ▶ Table view does not necessarily fill the entire view controller

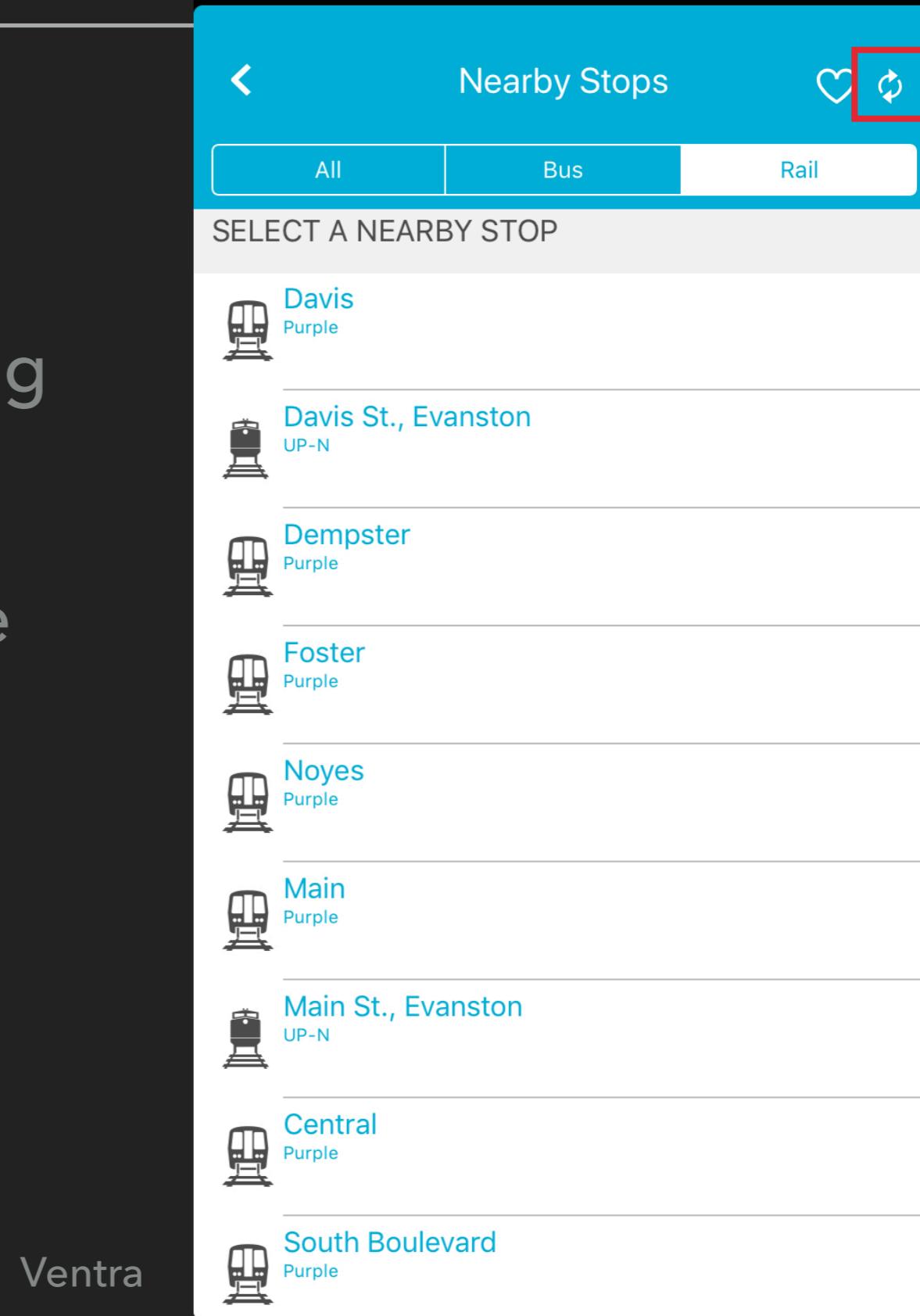


PULL-TO-REFRESH

PULL-TO-REFRESH

REFRESH BUTTON

- ▶ This is the old method of allowing users to refresh a page
- ▶ Refresh button takes up valuable real estate in the navigation bar



9:41



PULL-TO-REFRESH

PULL-TO-REFRESH

- ▶ Invented by Loren Brichter during the development of Tweetie
- ▶ Now a built-in component

Podcasts

Library

Sort

Episodes



.... DEC 19, 2018
 'Iraq's Post-ISIS Campaign of Revenge'
 'New Yorker' reporter Ben...



[Details](#) 48 min left



THURSDAY
 'Kennedy Vs. Carter & The Fight That Broke The Democratic Party'



[Details](#) 49 min



DEC 4, 2018
 'Remaking Journalism' In An Age Of Information Chaos



[Details](#) 50 min



JANUARY 11
 'The Sopranos' - 20th Anniversary Show
 Critic David Bianculli offers...



[Details](#) 49 min



'Iraq's Post-ISIS Campaign...



[Listen Now](#)



[Library](#)



[Browse](#)



[Search](#)

A BRIEF INTRODUCTION TO

NETWORKING

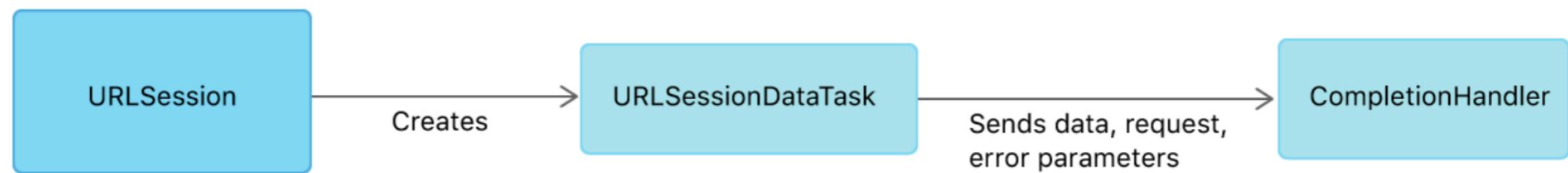
URLSESSION

- ▶ Almost all apps need to fetch data from the network
- ▶ This can be achieved using **URLSession** and related classes

URLSESSION

- ▶ Create a **URLSessionDataTask** by providing:
 - ▶ The URL for the resource you want to fetch
 - ▶ A completion handler (closure) that will be called when the data is available

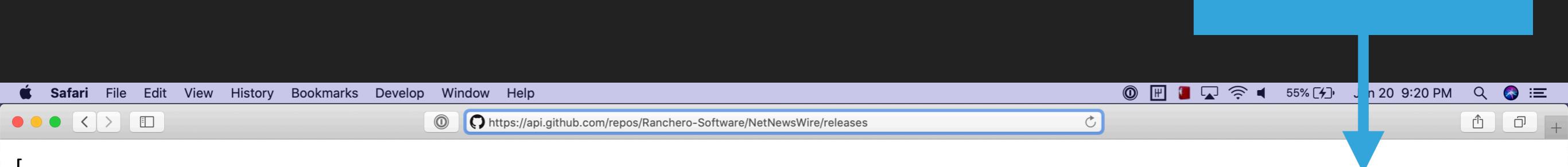
Figure 1 Creating a completion handler to receive results from a task



URLSESSION

- ▶ It's common for URLs to return data as JSON
- ▶ The app must parse the JSON into Swift model objects

GitHub API



```
[  
 {  
 "url": "https://api.github.com/repos/Ranchero-Software/NetNewsWire/releases/20888546",  
 "assets_url": "https://api.github.com/repos/Ranchero-Software/NetNewsWire/releases/20888546/assets",  
 "upload_url": "https://uploads.github.com/repos/Ranchero-Software/NetNewsWire/releases/20888546/assets{/name,label}",  
 "html_url": "https://github.com/Ranchero-Software/NetNewsWire/releases/tag/mac-5.0.3",  
 "id": 20888546,  
 "node_id": "MDc6UmVsZWFzZTIwODg4NTQ2",  
 "tag_name": "mac-5.0.3",  
 "target_commitish": "mac-release",  
 "name": "NetNewsWire 5.0.3",  
 "draft": false,  
 "author": {  
 "login": "brentsimmons",  
 "id": 1297121,  
 "node_id": "MDQ6VXNlcjEyOTcxMjE=",
```

JSON TO SWIFT

- ▶ **Decodable** is a protocol that indicates that a new instance of the type can be created by decoding JSON
 - ▶ Properties that are present in the JSON, but not the model are ignored
 - ▶ Properties that are present in the model, but potentially absent from the JSON must be optional

GITHUB API MODELS

```
2
3  struct Release: Decodable {
4      let name: String
5      let createdAt: String
6      let author: Author
7  }
8
9  struct Author: Decodable {
10     let login: String
11 }
12
```

GITHUB CLIENT

- ▶ GitHubClient is responsible for:
 - ▶ Fetching JSON from the GitHub API's /releases endpoint
 - ▶ Parsing the JSON into Swift model objects

```
14  
15  class GitHubClient {  
16  
17      static func fetchReleases(completion: @escaping ([Release]?, Error?) -> Void) {  
18          // TODO: - fetch a list of releases  
19      }  
20  }  
21
```

GITHUB CLIENT

- ▶ The **static** keyword indicates that this method lives on the type itself (not an instance of the type)

```
14  
15  class GitHubClient {  
16  
17      static func fetchReleases(completion: @escaping ([Release]?, Error?) -> Void) {  
18          // TODO: - fetch a list of releases  
19      }  
20  }  
21
```

GITHUB CLIENT

- ▶ **fetchReleases** takes a closure called **completion** as a parameter

```
14  
15  class GitHubClient {  
16  
17      static func fetchReleases(completion: @escaping ([Release]?, Error?) -> Void) {  
18          // TODO: - fetch a list of releases  
19      }  
20  }  
21
```

GITHUB CLIENT

- ▶ `@escaping` means this closure may be execute **after** the `fetchReleases` method returns

```
14  
15  class GitHubClient {  
16  
17      static func fetchReleases(completion: @escaping ([Release]?, Error?) -> Void) {  
18          // TODO: - fetch a list of releases  
19      }  
20  }  
21
```

GITHUB CLIENT

- ▶ First, we create a variable for the URL

```
14  
15 class GitHubClient {  
16  
17     static func fetchReleases(completion: @escaping ([Release]?, Error?) -> Void) {  
18         let url = URL(string: "https://api.github.com/repos/Ranchero-Software/NetNewsWire/releases")!  
19  
20         // ...  
21  
22 }
```

GITHUB CLIENT

- ▶ Next, we create a `URLSessionDataTask` to fetch data from the URL

```
14  
15 class GitHubClient {  
16  
17     static func fetchReleases(completion: @escaping ([Release]?, Error?) -> Void) {  
18         let url = URL(string: "https://api.github.com/repos/Ranchero-Software/NetNewsWire/releases")!  
19  
20         let task = URLSession.shared.dataTask(with: url) { data, _, error in  
21             if let data = data, let releases = try? JSONDecoder().decode([Release].self, from: data), error == nil {  
22                 completion(releases, nil)  
23             } else {  
24                 completion(nil, error)  
25             }  
26         }  
27         task.resume()  
28     }  
29 }
```

GITHUB CLIENT

- ▶ Inside the data task's closure, **guard** that data has a value and error is nil
- ▶ If one of these is not the case, call the **fetchReleases** closure with nil and the error

```
14  
15 class GitHubClient {  
16  
17     static func fetchReleases(completion: @escaping ([Release]?, Error?) -> Void) {  
18         let url = URL(string: "https://api.github.com/repos/Ranchero-Software/NetNewsWire/releases")!  
19  
20         let task = URLSession.shared.dataTask(with: url) { data, _, error in  
21  
22             guard let data = data, error == nil else {  
23                 DispatchQueue.main.async { completion(nil, error) }  
24                 return  
25             }  
26  
27             do {
```

GITHUB CLIENT

- ▶ Use `JSONDecoder` to decode the JSON into an array of `Release` objects
- ▶ This can throw an error, so we use a `do-try-catch` block

```
21
22     guard let data = data, error == nil else {
23         DispatchQueue.main.async { completion(nil, error) }
24         return
25     }
26
27     do {
28         let decoder = JSONDecoder()
29         decoder.keyDecodingStrategy = .convertFromSnakeCase
30         let releases = try decoder.decode([Release].self, from: data)
31         DispatchQueue.main.async { completion(releases, nil) }
32     } catch (let parsingError) {
33         DispatchQueue.main.async { completion(nil, parsingError) }
34     }
35 }
```

GITHUB CLIENT

- ▶ Call `resume()` to start the task
- ▶ It's created in the "paused" state

```
21
22     guard let data = data, error == nil else {
23         DispatchQueue.main.async { completion(nil, error) }
24         return
25     }
26
27     do {
28         let decoder = JSONDecoder()
29         decoder.keyDecodingStrategy = .convertFromSnakeCase
30         let releases = try decoder.decode([Release].self, from: data)
31         DispatchQueue.main.async { completion(releases, nil) }
32     } catch (let parsingError) {
33         DispatchQueue.main.async { completion(nil, parsingError) }
34     }
35
36     task.resume()
37 }
38 }
```

GITHUB CLIENT

```
14
15 class GitHubClient {
16
17     static func fetchReleases(completion: @escaping ([Release]?, Error?) -> Void) {
18         let url = URL(string: "https://api.github.com/repos/Ranchero-Software/NetNewsWire/releases")!
19
20         let task = URLSession.shared.dataTask(with: url) { data, _, error in
21
22             guard let data = data, error == nil else {
23                 DispatchQueue.main.async { completion(nil, error) }
24                 return
25             }
26
27             do {
28                 let decoder = JSONDecoder()
29                 decoder.keyDecodingStrategy = .convertFromSnakeCase
30                 let releases = try decoder.decode([Release].self, from: data)
31                 DispatchQueue.main.async { completion(releases, nil) }
32             } catch (let parsingError) {
33                 DispatchQueue.main.async { completion(nil, parsingError) }
34             }
35
36             task.resume()
37         }
38     }
39 }
```

USING THE GITHUB CLIENT

```
40
41 GitHubClient.fetchReleases { (releases, error) in
42     guard let releases = releases, error == nil else {
43         print(error!)
44         return
45     }
46
47     for release in releases {
48         print("""
49             ✓ \(release.name)
50             Author: \(release.author.login)
51             Date: \(release.createdAt)
52
53             """)
54     }
55 }
56
```

UPDATES TO THE UI SHOULD
ALWAYS HAPPEN ON THE MAIN
QUEUE

Remember this for interviews.

ASSIGNMENT #3

GITHUB ISSUES

GITHUB ISSUES

- ▶ Build an app that lists open and closed GitHub issues from an open-source repo
- ▶ We will use:
 - ▶ UITabBarController
 - ▶ UINavigationController
 - ▶ UITableViewcontroller
 - ▶ URLSession

GITHUB ISSUES

- ▶ Due Wednesday, January 29 at 5:29pm
- ▶ Post any questions to the class forum
- ▶ We're getting into more complex material, but please refrain from using any 3rd-party frameworks

GITHUB ISSUES

- ▶ Remember to commit to the **development** branch and then open a pull request into master